Visuelle Perzeption für Mensch-Maschine Schnittstellen

Vorlesung, WS 2013 / 2014

Dr. Saquib Sarfraz
Prof. Dr. Rainer Stiefelhagen

Institut für Anthropomatik
Karlsruher Institut für Technologie - KIT

http://cvhci.anthropomatik.kit.edu/
rainer.stiefelhagen@kit.edu

Dense Feature Descriptors

WS 2013/14

saquib.sarfraz@kit.edu

Institute for Anthropomatics
Computer Vision for Human-Computer Interaction

Today

- What are Visual Descriptors
- How to compute them
- Dense features
- Dense Feature Encoding

Institute for Anthropomatics
Computer Vision for Human-Computer Interaction

Visual Descriptors

- Descriptors for image classification should be
  - Discriminative
  - Robust against image transformations
  - Robust against object transformations, viewpoint and occlusion
  - Efficient to compute

Visual Descriptors

- Color Descriptors
- Texture Descriptors
- Local Descriptors

Conclusion from Color Matching

- Three primaries are sufficient for most people to reproduce arbitrary colors
  - The human eye normally contains only three types of color receptors, called cone cells
  - Each color receptor responds to different ranges of the color spectrum
  - Humans respond to the light stimulus via a three dimensional sensation, which generally can be modeled as a mixture of three primary colors

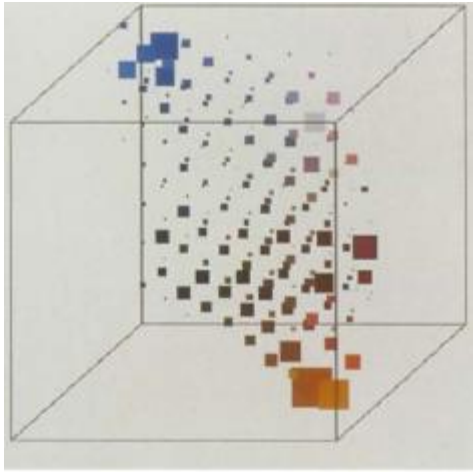
Color Models

- Different ways of parameterizing 3D color space, e.g.
  - RGB
  - XYZ
  - CMY
  - HSV
  - …

Color Histogram

- A color histogram represents the distribution of colors where each histogram bin corresponds to a color in the quantized color space
- Color histogram as feature descriptor
  - Color space selection
  - Color space quantization
  - Histogram computation
  - Histogram distance metrics

Color Histogram



Pros and Cons

- Pros
  - Easy and fast to compute
  - Compact representation of color information
  - Can easily be normalized so that different image histograms can be compared
- Cons
  - Local information (not able to extract spatial localized feature)
  - fixed-size structures, cannot achieve a balance between expressiveness and efficiency

Color Moments

- Central moments are statistics
  - First order = mean
  - Second order = variance
  - Third order = skew
  - Fourth order = kurtosis
  - High order moments are less intuitive

$$m_d = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^d$$

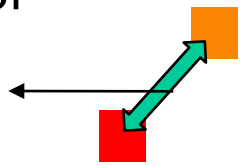
$$m_1 = 132.4$$
$$m_2 = 2008.2$$
$$m_3 = 4226$$
$$m_4 = 12.6 \times 10^6$$

- For color images, take moments of each band

Color Correlogram

- Describe global distribution of local spatial correlation of colors
- A table indexed by color pairs, $\mathrm{P}(c_i, c_j, d)$ specifies the probability of finding a pixel of color c_j at a distance *d* from a pixel of color c_i in the image

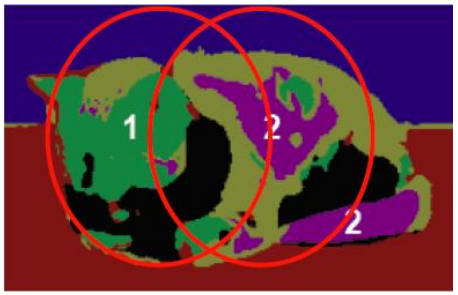
  e.g. P (Red, Orange, 4.42) = Probability of



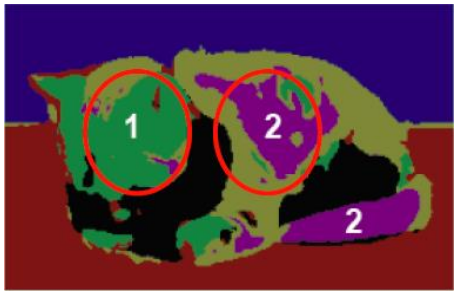
4.42

- An *Autocorrelogram* captures spatial correlation between identical colors => subset of correlograms

Circular kernel Correlogram

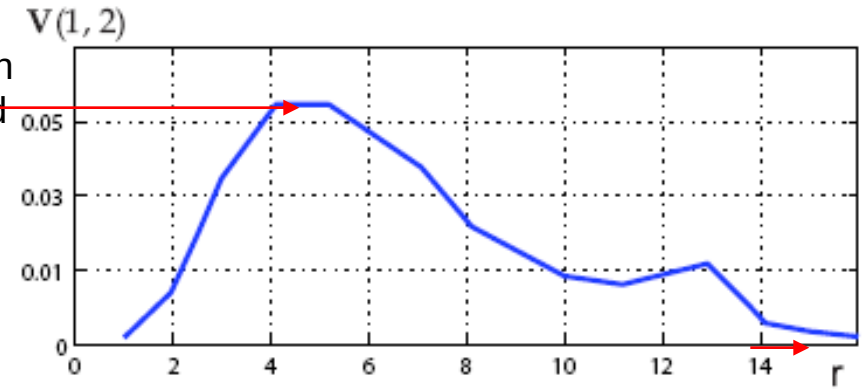


radius r = distance between regions



other radii

Correlation = pixel match between two regions covered by kernel



Maximum Correlation: (radius of kernel r = distance between regions)

Correlation decreases as difference between radius and distance increase

Properties of Correlogram

- Invariant to translation
- Circular kernels induce rotational invariance
  - Rectangular kernels computationally more efficient
- Robuts to affine, perspective transformations
- Robust to general object pose changes

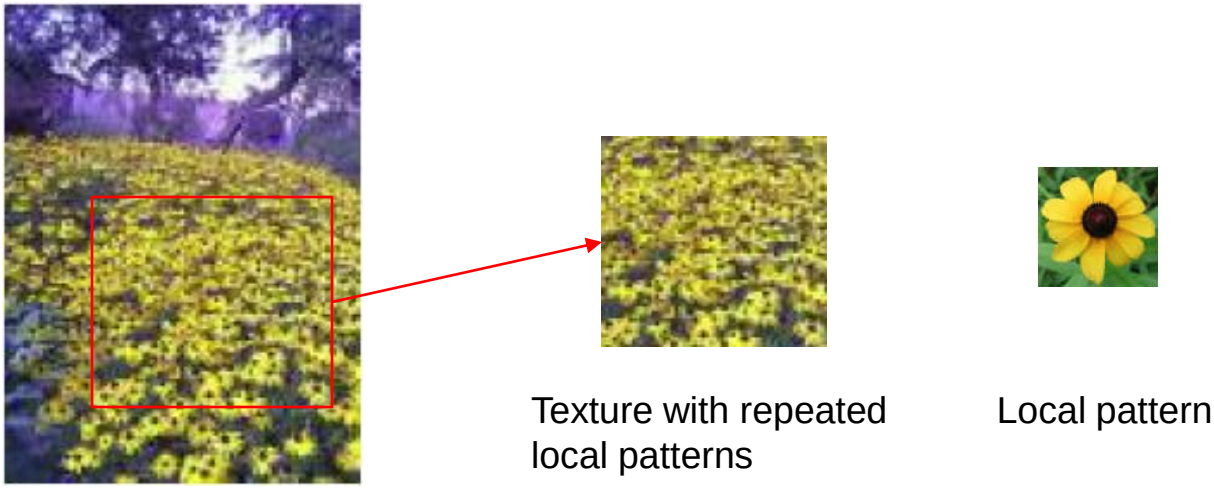
- Not invariant with respect to scale! => learn with multiple training images at multiple scales

- Can be computed in $O(K*N^2)$, $K<<N$

Texture

What is Texture?

- Often used to represent all the “details” in the image
- One or more basic local patterns that are repeated in a periodic manner

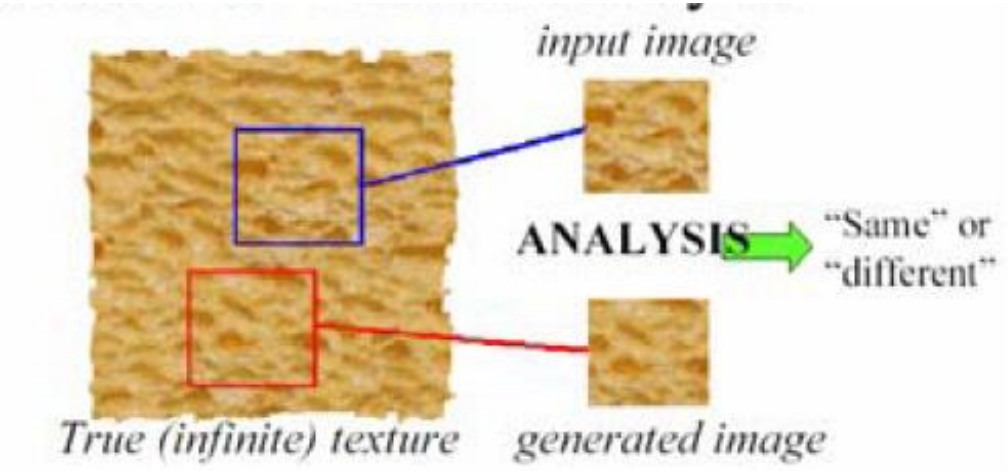


Texture with repeated local patterns

Local pattern

Discrimination

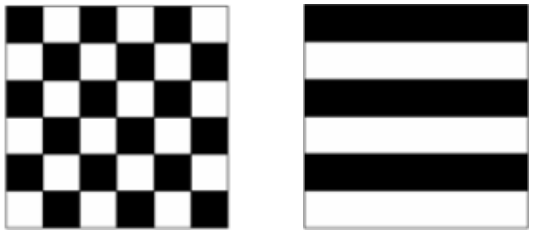
- Goal of texture analysis



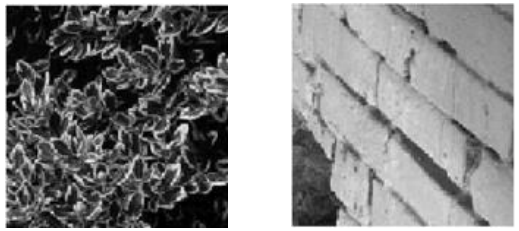
  - Compare textures and decide if they're made of the same thing

Texture Analysis

- Two approaches for texture analysis:
  - Structural (top down)
    - Decompose image into basic elements or texels (textons)
    - Convenient for artificial textures
  - Statistical (bottom up)
    - Texture is property that can be derived from the statistics of small group of pixels, such as mean and variance
    - Convenient for natural textures



Artificial textures

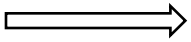


Natural textures

Statistical Approach

- Not always easy to define and segment out texels, especially for natural scenes





Look like similar, but difficult to see a texel structure; might have similar statistics

- Numeric quantities or statistics that describe a texture can be computed from the gray level values (or colors) alone
- Less intuitive, but computationally efficient
- It can be used for both classification and segmentation

- Alternatives:
  - Co-occurrence matrices
  - Edge histogram
  - Wavelets
  - etc…

Texture Models

- Transform an image window into a set of numbers (feature vector)
- Patches of the same texture should cluster in feature space

- Textures are made up of repeated subelements, with similar statistical properties
- Problem: find the subelements and represent their statistics

Identifying subelements

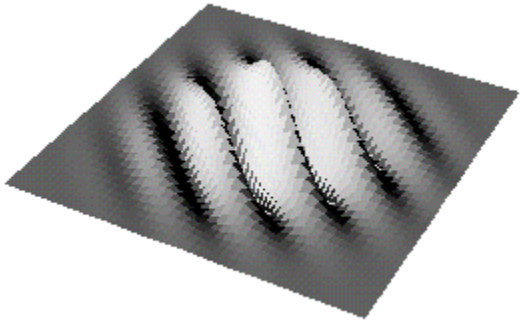
- Look for specific shapes
  - But: no known canonical set of textons
- Use a set of filters that capture simple pattern elements
- Human vision suggests spots and oriented filters at different scales
  - Spots: typically symmetric Gaussians
  - Bars: typically oriented Gaussians

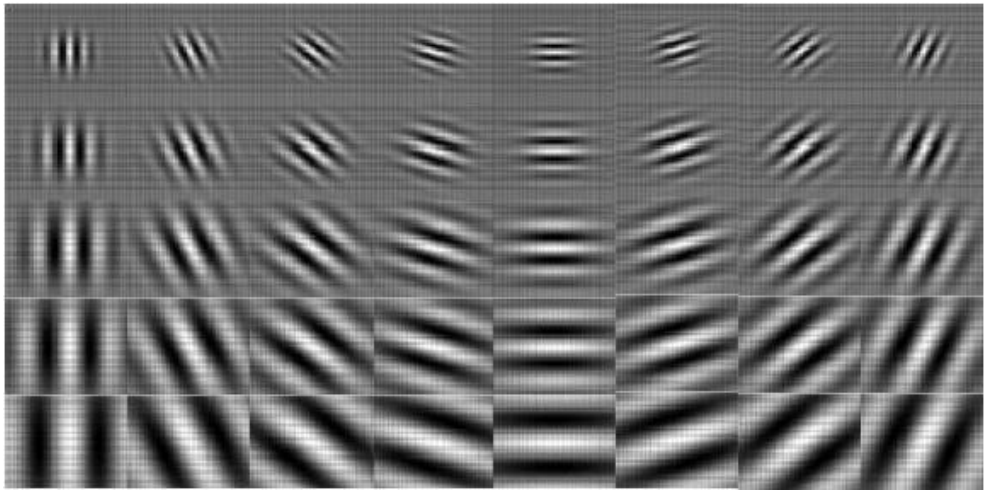
Choice of Filters

- No obvious advantage to any type of oriented filters
  - Weighted sum of Gaussians
  - Gabor filters
  - Wavelets
- How many filters?
  - Literature suggests 4-11 scales and 2-18 orientations (typically 6 orientations suffice)
- Tradeoff: detail of representation versus cost of computation

Gabor Filters

- 2-D sine waves modulated by a Gaussian envelope.
- Good models of the receptive fields found in simple cells of the primary visual cortex.
- 2D Gabor filter at different scales and orientations (spatial domain):





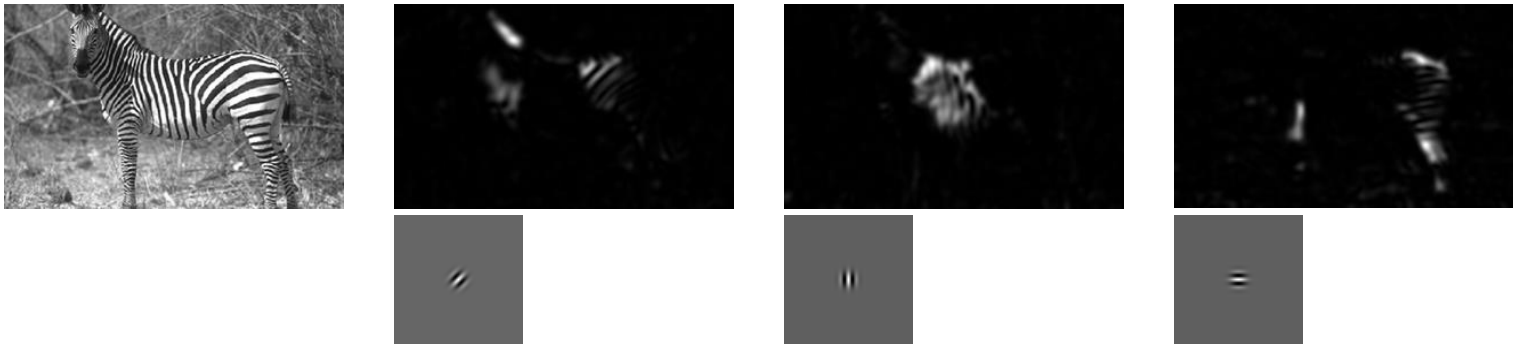
Typical: 5 scales, 8 orientations

Gabor Wavelet Transform

- For a given image *I(x,y)*, perform convolution with gabor filter kernels

$$G_{mn}(x,y) = \sum_{s=0}^{S}\sum_{t=0}^{T} I(x-s, y-t) g_{m,n}(s,t)$$

- Compute magnitude with real and imaginary part responses



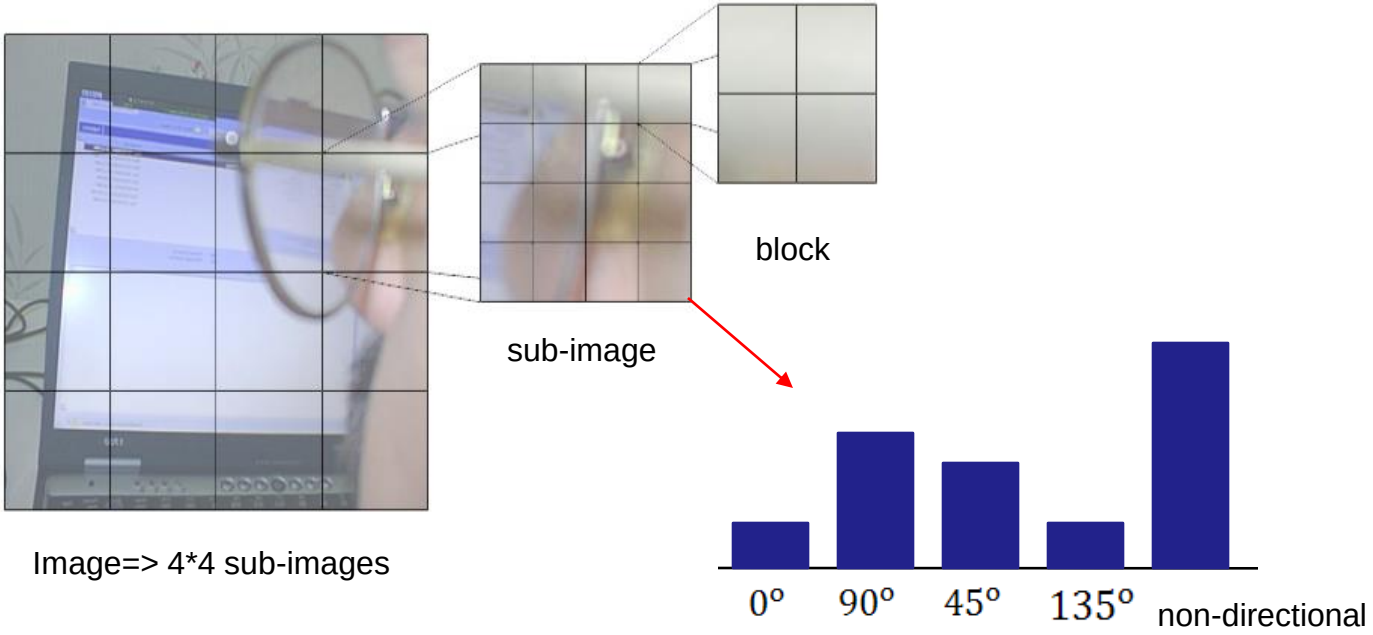
Example filter response in magnitude with corresponding Gabor filter (real part)

Other Models: Edge Histogram

- Captures the spatial distribution of different types of edges
  - Partition image into large local regions
  - Partition each local region into small image patches
  - Quantize each patch as horizontal, vertical, diagonal
    - Use gradient filters
  - Collect results into local region histograms
  - Perform matching based on histograms

Edge Histogram Descriptor

- Edge Histogram Descriptor (EHD) in MPEG-7 for calculating frame similarity



Image=> 4*4 sub-images

A 5-bin histogram in for each sub-image

Spatial Information

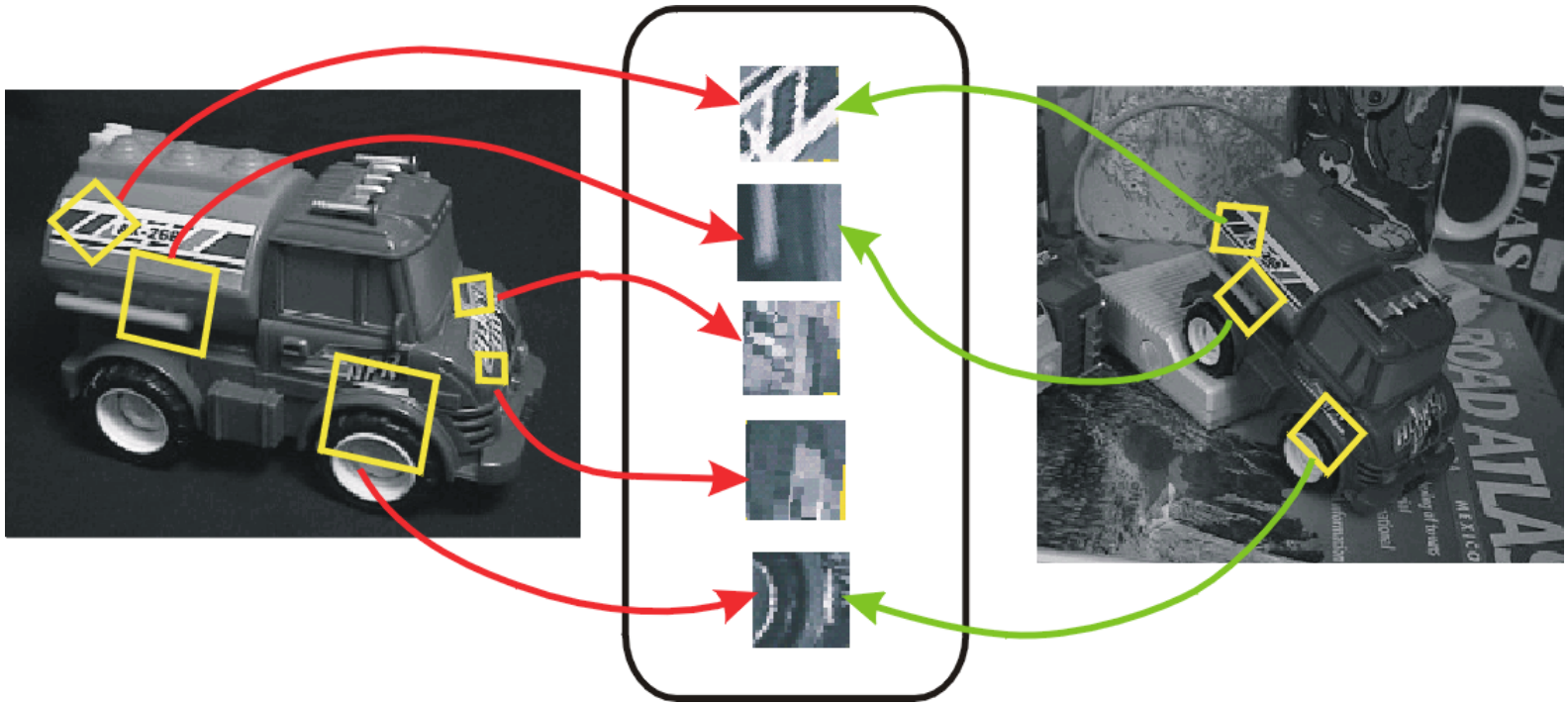
- In which image areas should the descriptors be computed?
  - Whole image?
  - Sub-windows?
- Some spatial information can be kept by extracting descriptors on sub-windows
- Other approaches
  - Local descriptors / key point detection

…

- Local Descriptors

Invariant Local Feature

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters

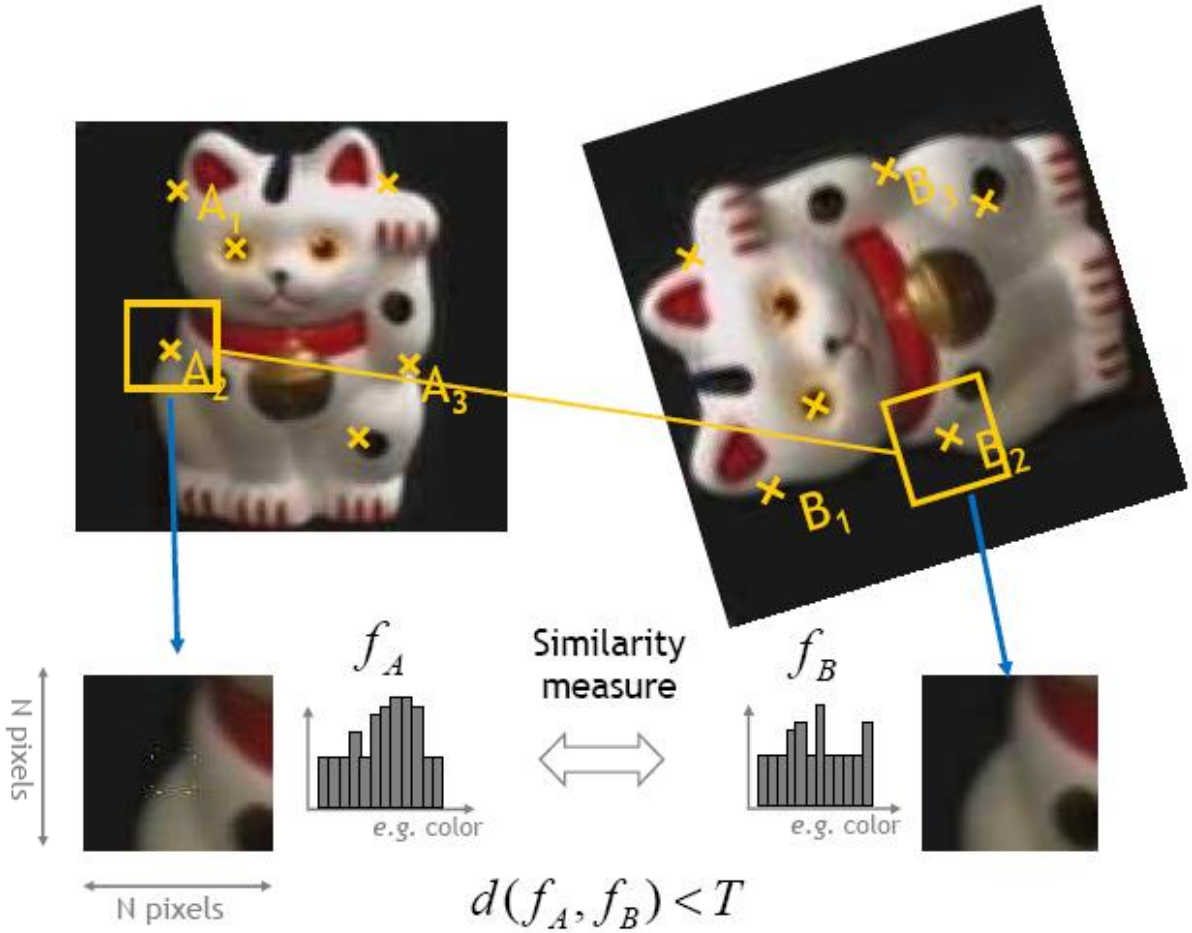


local feature patches

Components of local feature

- Key or interest points
  - Specify repeatable points
  - x-,y-position and scale
  - e.g. corners, blobs
- Local (key point) descriptors
  - Define the feature representation around an interest point
  - e.g raw pixels or a histogram of gradient in the neighborhood of a key point

Approach

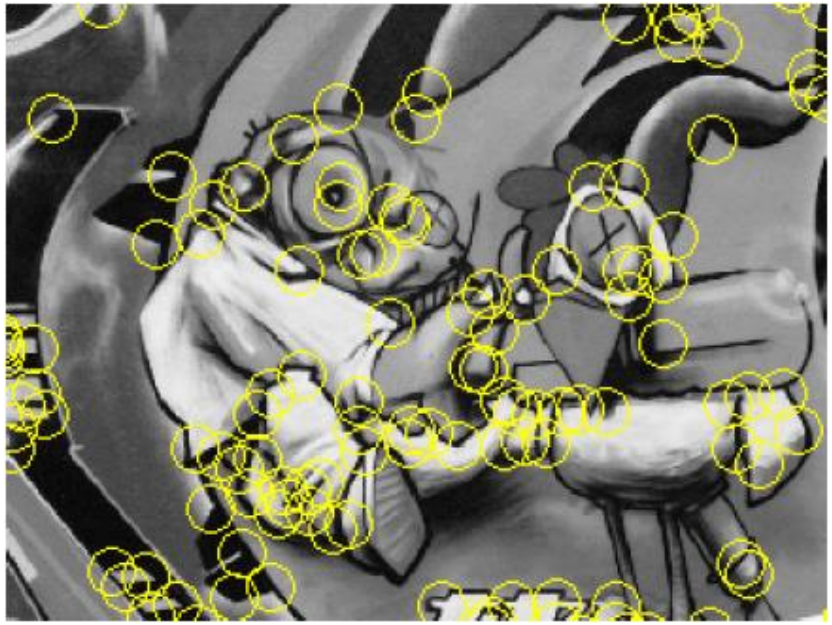


1. Find a set of distinctive key-points
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

Keypoint Detectors

- Many existing detectors available
  - Hessian & Harris [Beaudet ‘78],[Harris ‘88]
  - Laplacian, DoG [Lindeberg ‘98],[Lowe ‘99]
  - Harris-/Hessian-Laplace [Mikolajczyk & Schmid ‘01]
  - Harris-/Hessian-Affine [Mikolajczyk & Schmid ‘04]
  - EBR and IBR [Tuytelaars & Van Gool ‘04]
  - MSER [Matas ‘02]
  - Salient Regions [Kadir & Brady ‘01]
  - Dense Sampling
  - Others…
- Reference site:
  - http://www.robots.ox.ac.uk/~vgg/research/affine/index.html

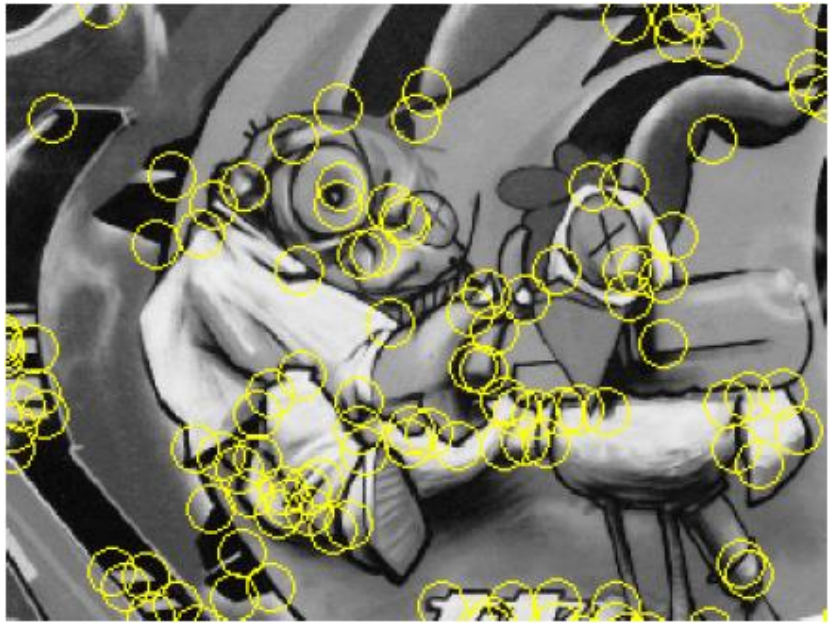
Keypoint Localization



- Goals:
  - Repeatable detection
  - Precise localization
  - Interesting content

  => *Look for two-dimensional signal changes*

Keypoint Localization



- Goals:
  - Repeatable detection
  - Precise localization
  - Interesting content

  => *Look for two-dimensional signal changes*

Local Descriptor : SIFT

- The area around the keypoint is divided into 4 x 4 subregions
- Build an orientation histogram with 8 bins for each subregion; gradient values are weighted by a Gaussian window
- This results in a vector with 128 dimensions (4 x 4 x 8)
- Normalize this vector to unit length (grants invariance to multiplicative changes in lighting)

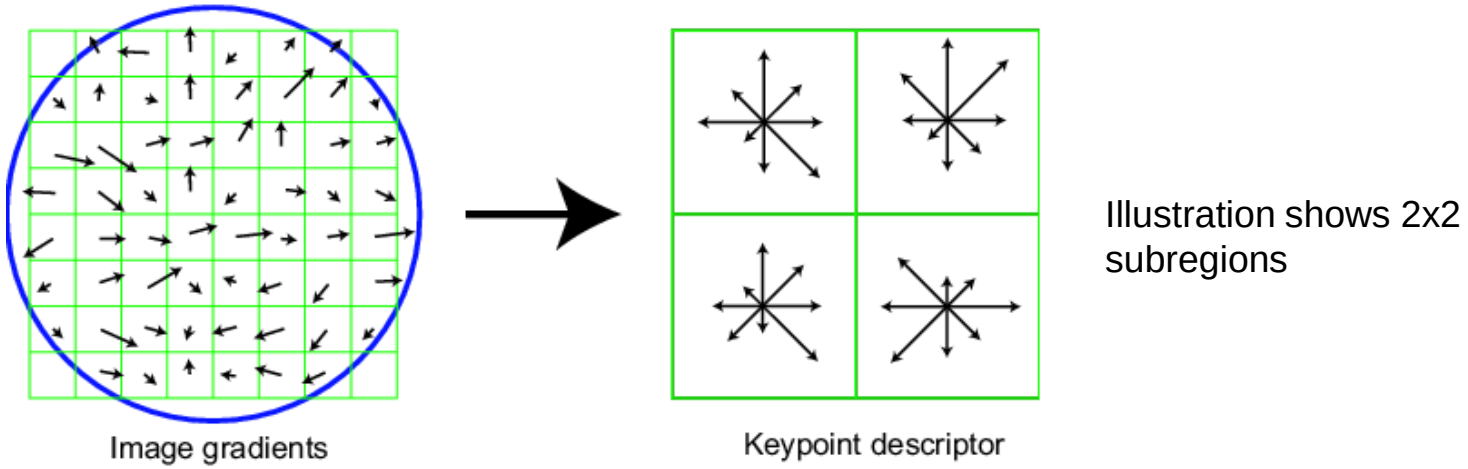
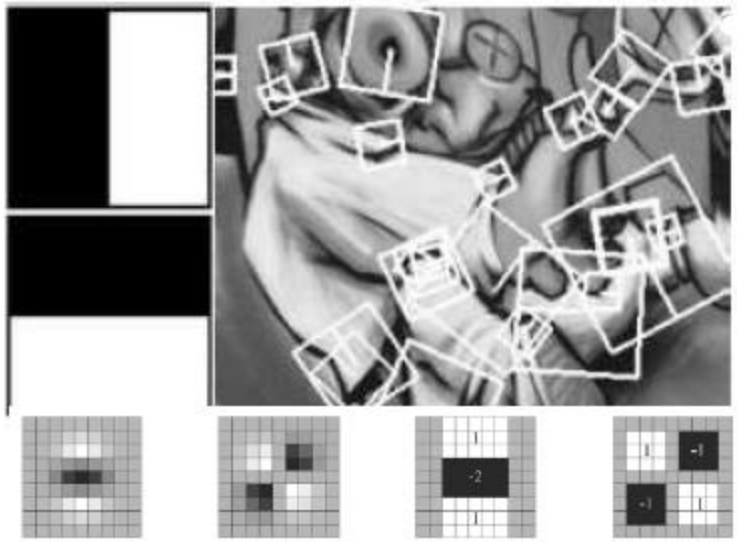


Illustration shows 2x2 subregions

SIFT-Features: Properties

- Scale-invariant
- Rotation-invariant
- Robust to illumination change
- Robust to noise
- Robust to minor changes in view-point

Local Descriptor : SURF

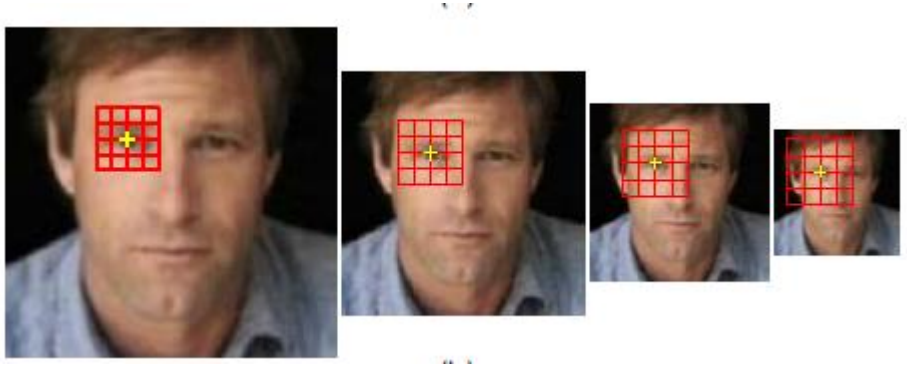


- Fast approximation of SIFT idea
  - Efficient computation by 2D box filters & integral images ⇒ 6 times faster than SIFT
- Equivalent quality for object identification
- GPU implementation available
  - Feature extraction @ 100Hz (detector + descriptor, 640×480 img) http://www.vision.ee.ethz.ch/~surf

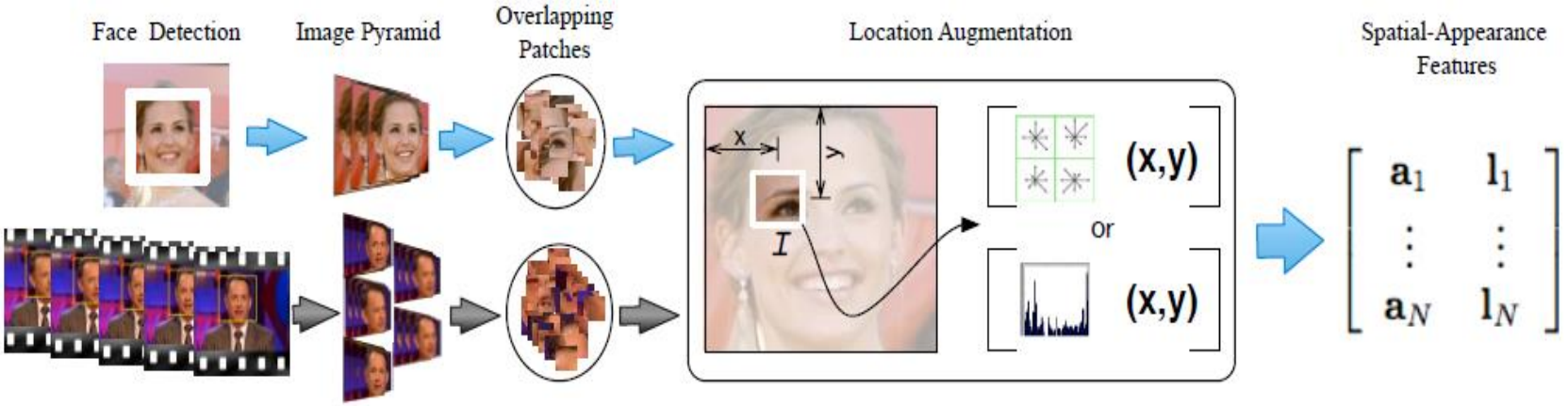
[Bay, ECCV'06], [Cornelis, CVGPU'08]

Dense Features

Computing features densly
For example on overlapping patches in many scales in the image.



Problem: very very high dimensionality



No. Of feature vectors Very Large

e.g. Using a 24x24 pixel patch with 1-pixel stride on a 160 x 125 image in 5 scales
Give rise to 26000 d-dimensional feature vectors per image.

That means

If we want to represent an image by stacking these together into one vector

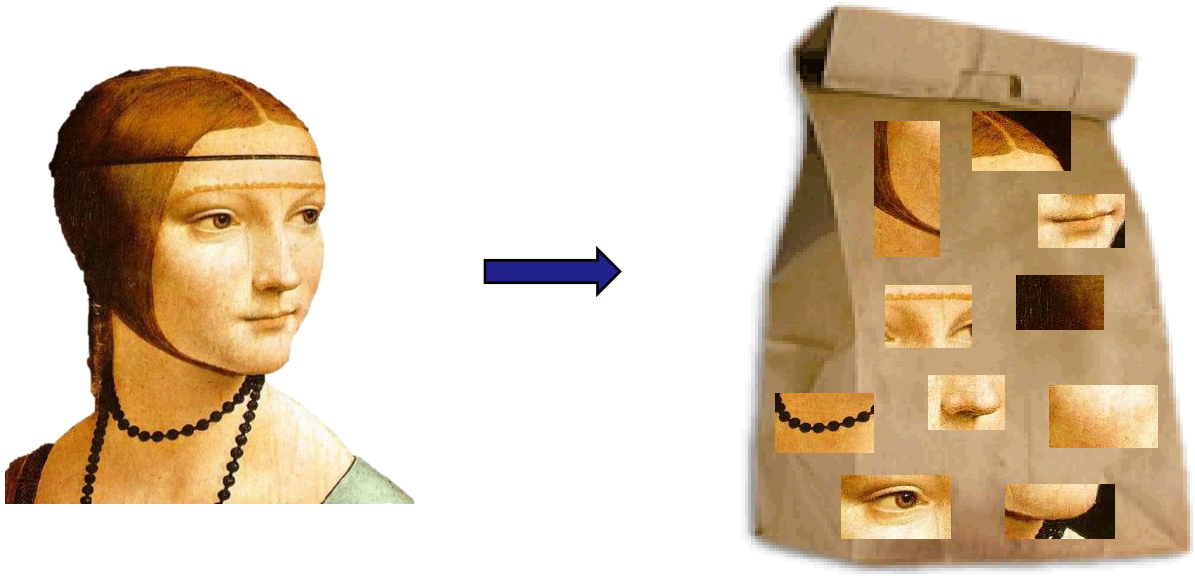
The resulting dimensionality would be

26000 x128 ~ 3.3 million dimensional vector

Solution: Encode into a compact form

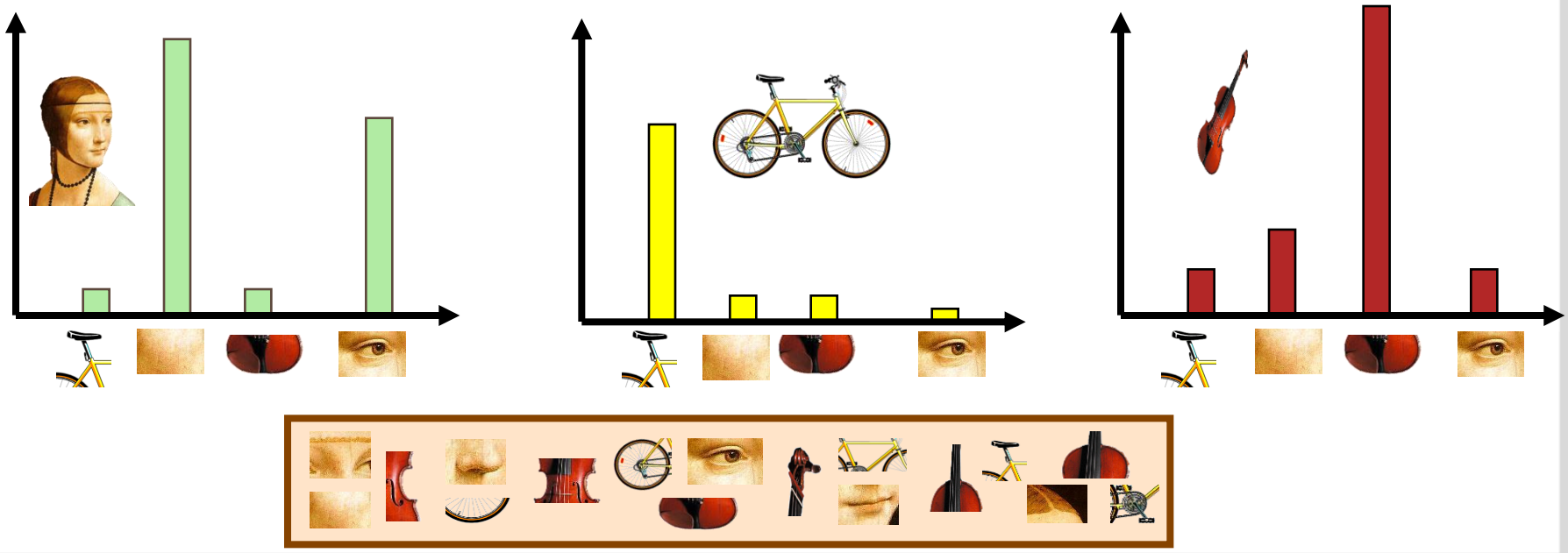
- Bag of Visual Word (BoVW) Model
- Fisher encoding

Bag-of-words



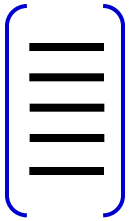
Bag-of-Words

- Analogy to text documents
- Definition
  - Independent features
  - histogram representation



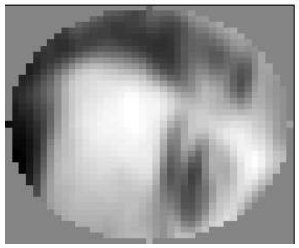
Build Visual-word Vocabulary-1

- Detect feature : Regular grid or Interest point
- Represent with local descriptor, e.g. SIFT

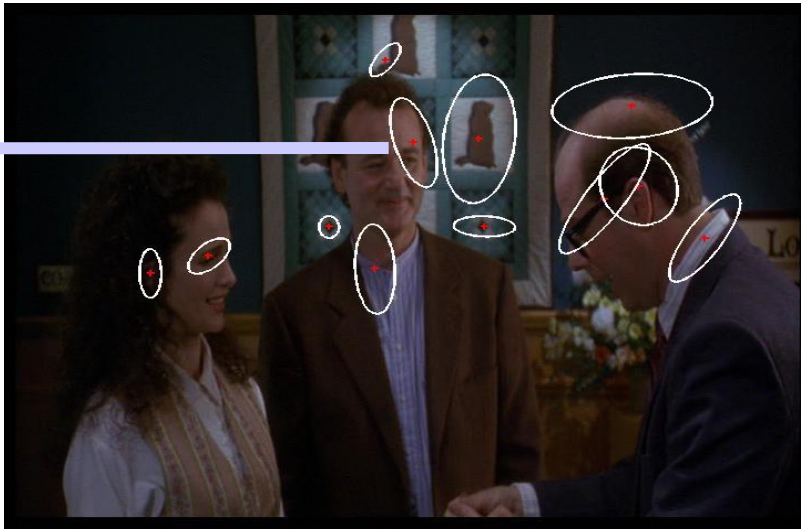


Compute SIFT descriptor

[Lowe'99]



Normalize patch



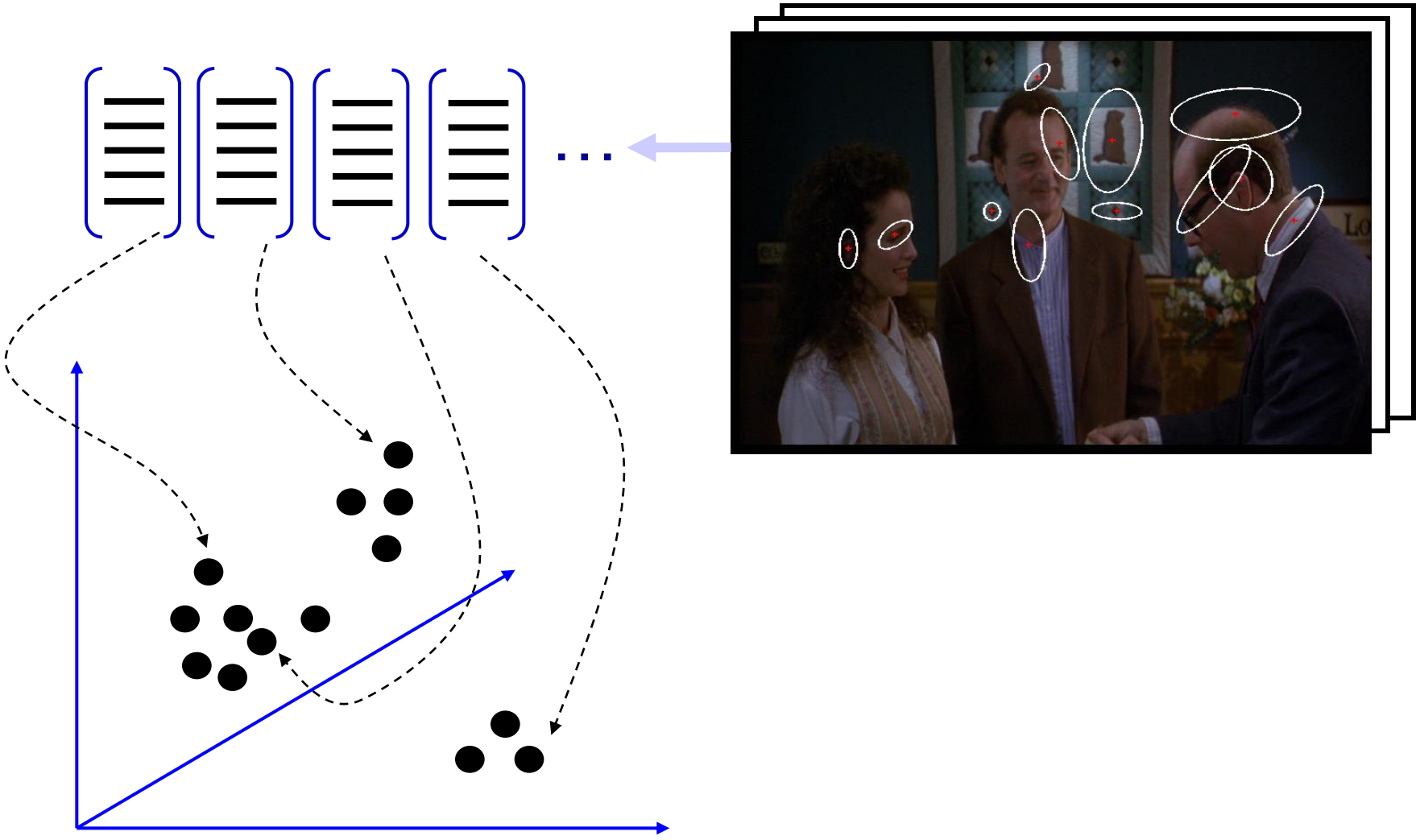
Detect patches

[Mikojaczyk and Schmid '02]

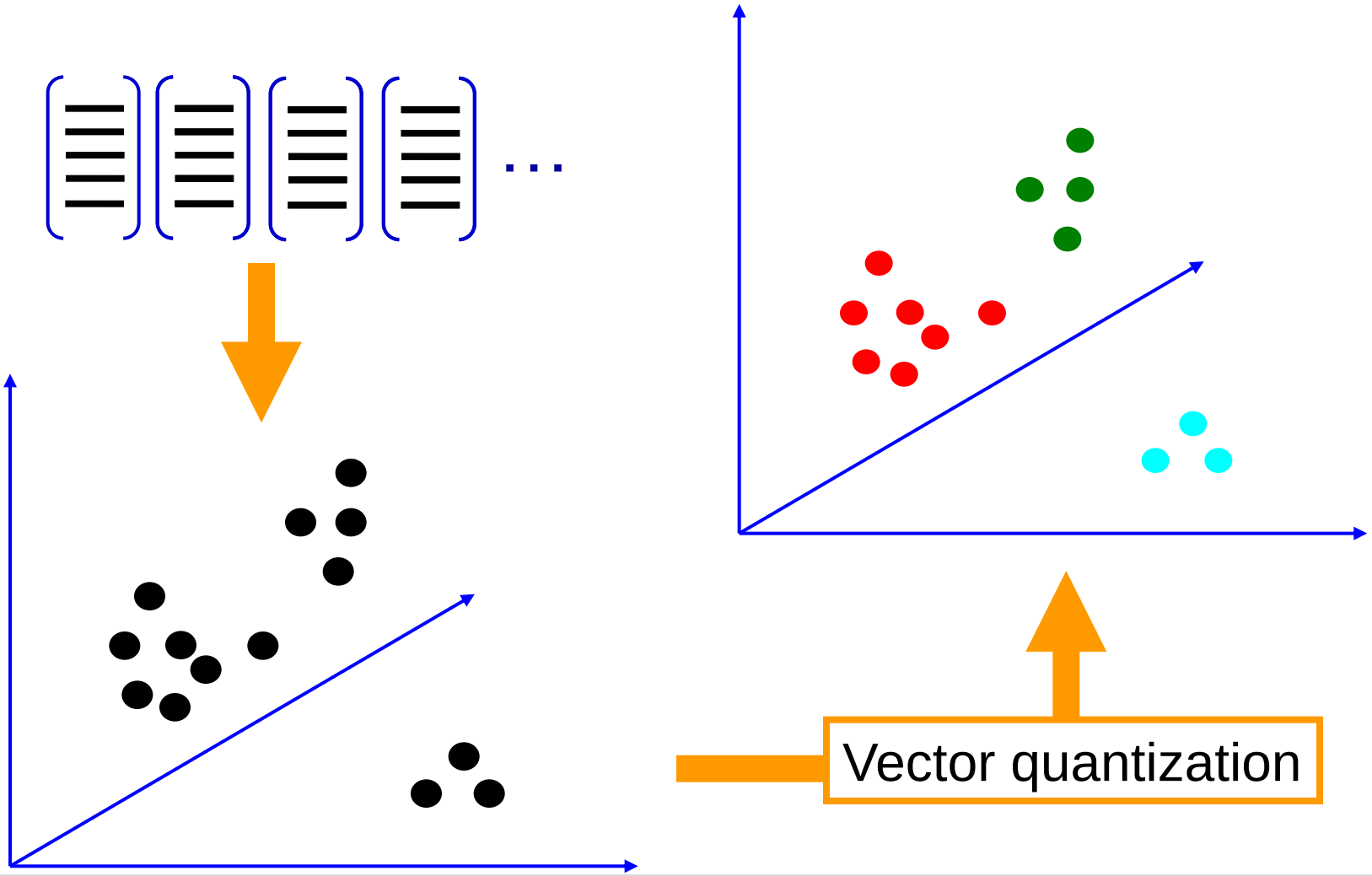
[Mata, Chum, Urban & Pajdla, '02]

[Sivic & Zisserman, '03]

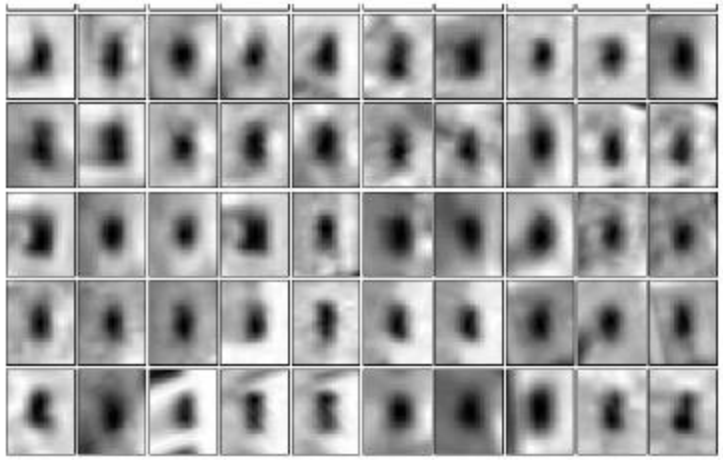
Build Visual-word Vocabulary-2

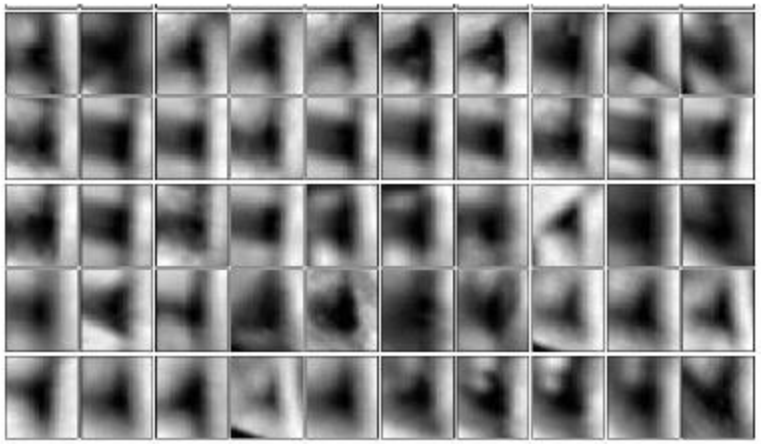


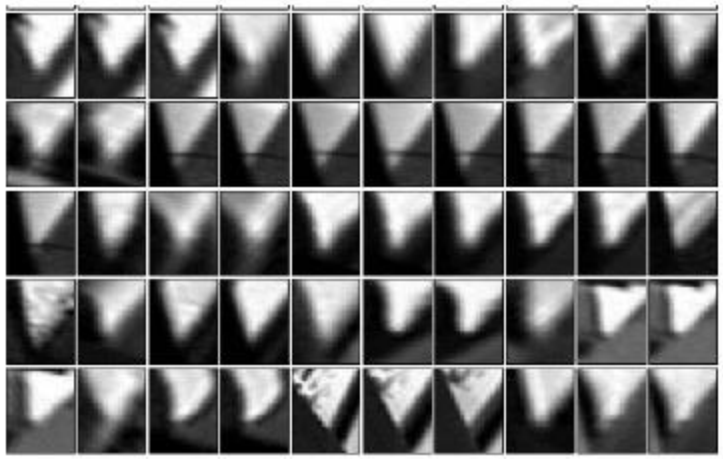
Build Visual-word Vocabulary-3

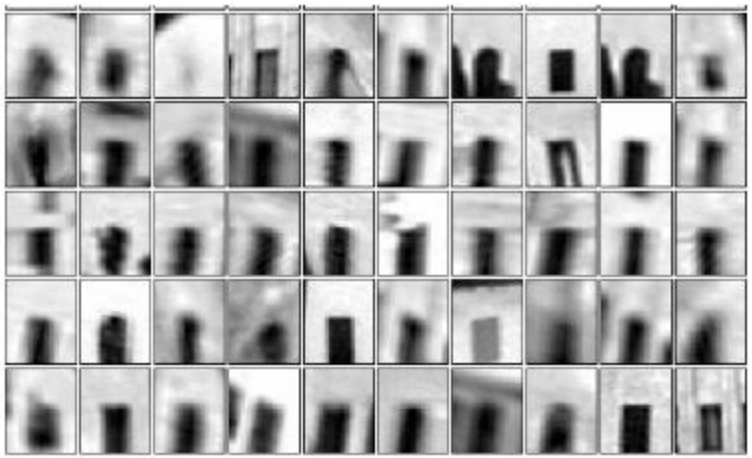


Samples of visual words



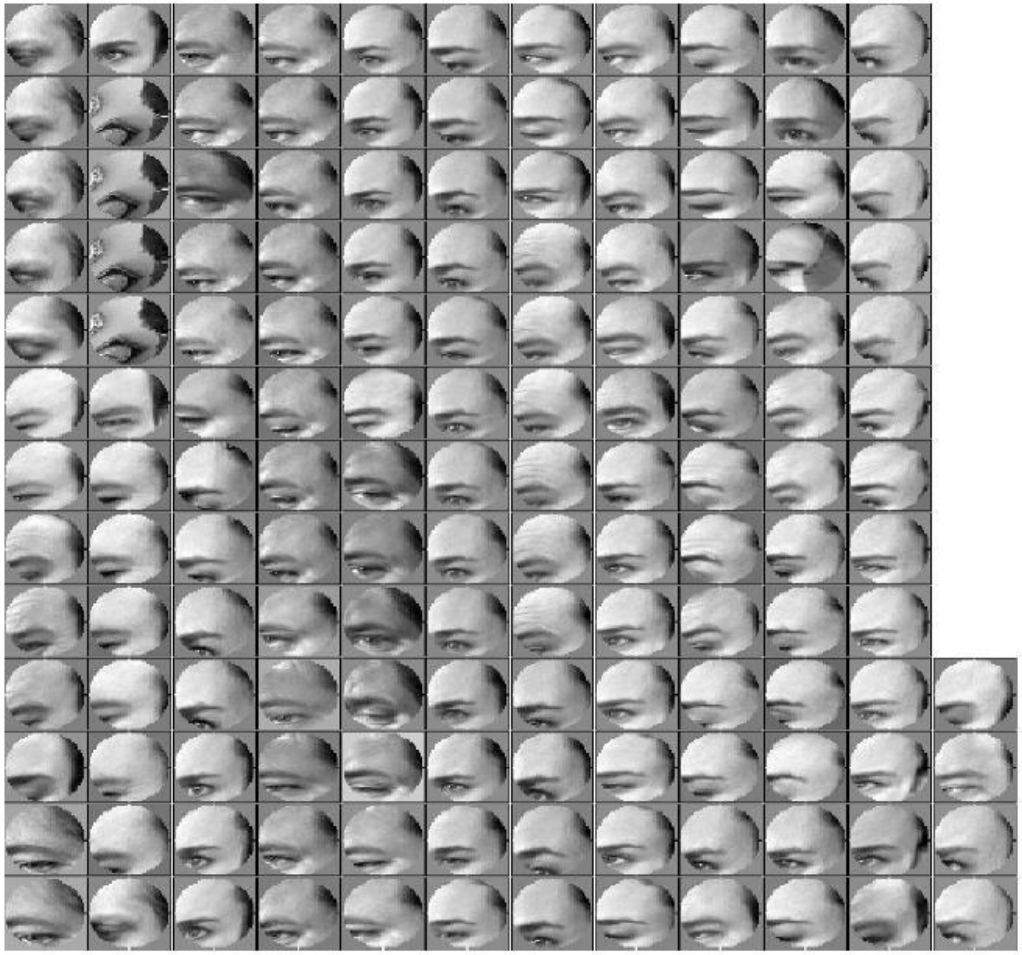


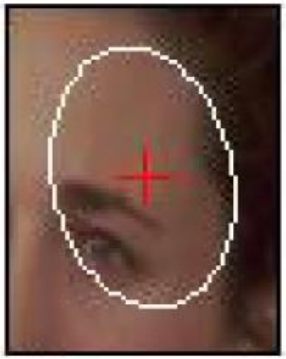




Clusters on SIFT Descriptors

Samples of visual words





Clusters on SIFT Descriptors : specific feature

Image patch examples of visual words

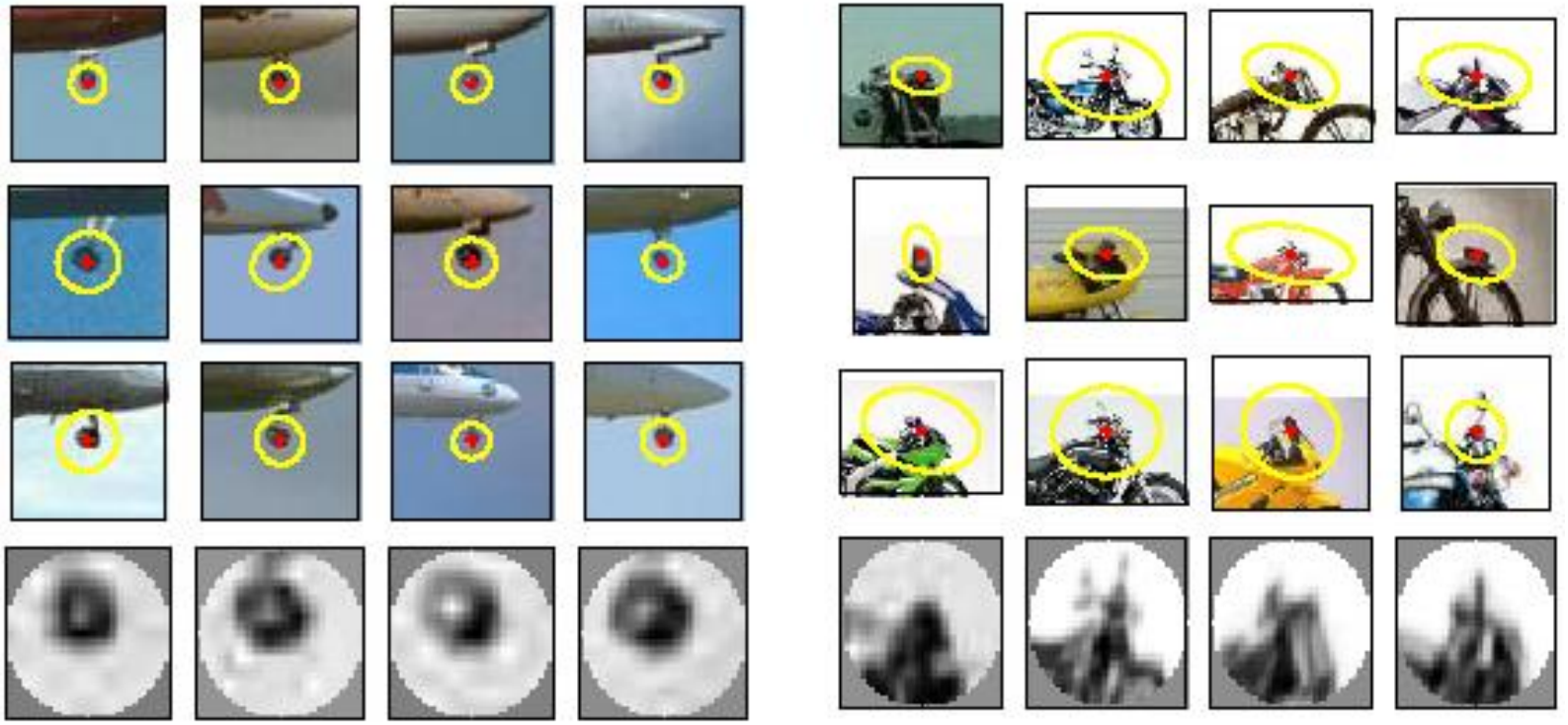
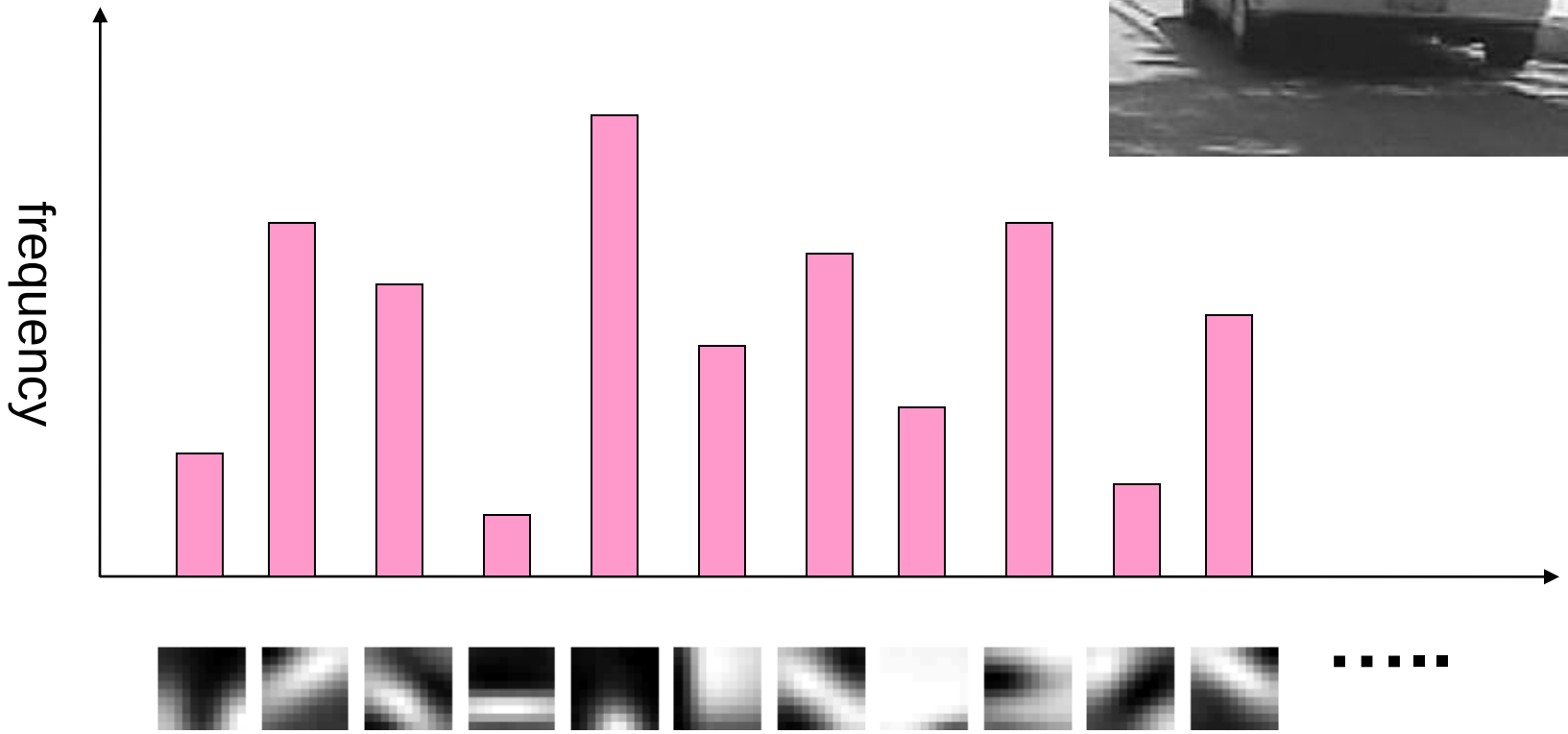
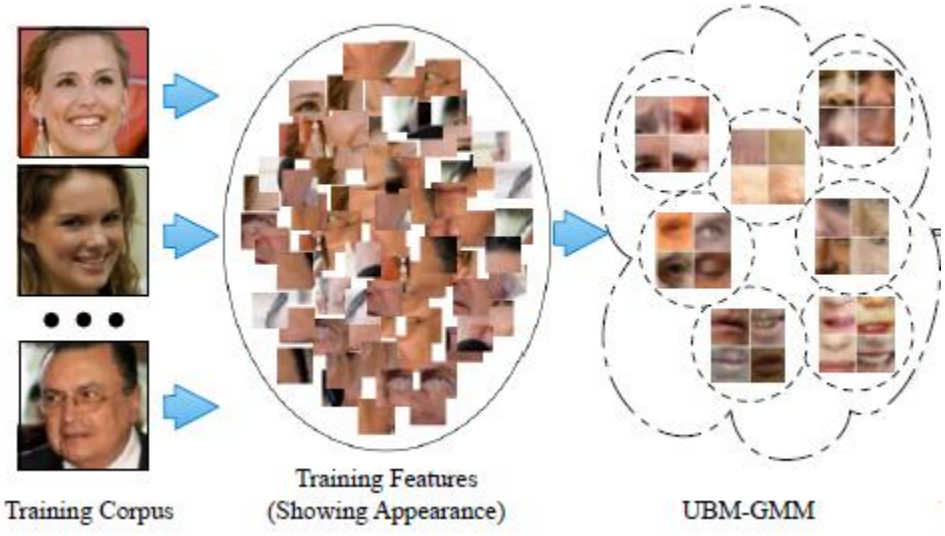


Image Representation



Fisher Encoding

- Aggregates feature vectors into a compact representation.
- Fitting a parametric generative model e.g. Gaussian Mixture Model GMM
- Encode derivative of the likelihood of Model with respect to its parameters.



Captruing the average first and second order differences between dense features and each of GMM centers.

$$\Phi_k^{(1)} = \frac{1}{N\sqrt{w_k}} \sum_{p=1}^{N} \alpha_p(k) \left(\frac{x_p - \mu_k}{\sigma_k} \right), \quad \Phi_k^{(2)} = \frac{1}{N\sqrt{2w_k}} \sum_{p=1}^{N} \alpha_p(k) \left(\frac{(x_p - \mu_k)^2}{\sigma_k^2} - 1 \right)$$

A Fisher Vector is obtained by stacking these difference vectors

$$\phi = \left[\Phi_1^{(1)}, \Phi_1^{(2)}, \ldots, \Phi_K^{(1)}, \Phi_K^{(2)}\right]$$

The Fisher Vectors dimensionality is 2 x K x d , where d is the dimensionality of the underlying featue vectors.

For example,
If we use SIFT (128-dim) in the dense feature computation, and train a 512 component GMM. The resulting Fisher vector would be 130,000 dimensional.

For our previous example (3.2 milion dimentional vector), its a lot of compactness ☺

Dimnetionalty still high ???

We can use some subspace learning to compress it further

e.g using subspace methods....

PCA ---- not a good idea

There exist some nice subspace learning methods that can take it down nicely e.g. From 130,000 dimensions to 1000 dimentions without any loss of accuracy.